Research Article

Optimization of Simultaneous Delivery and Pickup Wagon Scheme on Hybrid Siding Network of Railway Terminal

Bing Li, Ce Yun, Guanglan Xu, Xiaoqian Zhou, and Hua Xuan

School of Management Engineering, Zhengzhou University, Zhengzhou 450001, Henan, China

Correspondence should be addressed to Bing Li; lbing@zzu.edu.cn

Received 22 January 2022; Revised 28 March 2022; Accepted 30 March 2022; Published 11 May 2022

Academic Editor: Giulio E. Cantarella

Copyright © 2022 Bing Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Aiming at the local freight train transit system in railway terminal, the optimization of simultaneous delivery and pickup wagon scheme on hybrid siding network is presented. The problem is formulated as a mathematical programming model. The objective function intends to minimize shunting engines’ operating cost, wagons’ travelling cost, and penalty cost for exceeding the retrieval time window. Some constraints indicating the relationship between delivery and pickup wagons, relationship between radial branches and operation batches, relationship between local wagon groups’ retrieved time and outbound train departure time, and relationship between tasks and operation batches are considered in the model. As an NP problem, using traditional method to solve the model is difficult and inefficient. A novel two-stage hybrid optimization procedure is proposed. Firstly, the three-phase approach composed of coding task sequence, dividing task batch, and generating access order sequence population is developed to generate initial solution. Secondly, an asynchronous iteration heuristic is provided. The iteration object is given as the expression form of the solution representing the delivery and pickup wagon sequence and then the initial iteration population is generated. Furthermore, the primary updating iteration population with pending and remaining groups is introduced to optimize iteration object Then, the senior updating iteration population with Lévy flight searching is also introduced to optimize further iteration object. Finally, the experimental scenarios are designed to test the proposed approach. Also, the proposed approach is compared with some other approaches and the performance of the proposed approach is evaluated by some different sized instances.

1. Introduction

Railway freight terminal is a huge and complex system which consists of marshalling yards, handling stations, and railway sidings. Railway freight terminal is the important component of the railway freight transportation chain. Therefore, it is crucial for enhancing the transportation chain effectiveness that the performance of railway freight terminal is optimized.

Delivery and pickup scheme of local wagons is a significant work for the large-scale railway freight terminals. The shunting efficiency of wagons is directly related to the wagons’ turnover rate, the goods’ delivery time, and the accomplishment of the transport production quota. The railway terminal operators must consider how to appropriately organize the delivery and pickup scheme of wagons. Throughout the whole process of wagon shunting, the local wagons will be detached sequentially from different inbound trains, delivered to relevant handling station for loading or unloading, and then retrieved back and attached onto assigned outbound trains. According to the spatial layout of handling stations in a railway terminal, we can divide these handling areas into three types: branch-shaped siding network (BSSN), radial siding network (RSN), and hybrid siding network (HSN). For a branch-shaped siding network, after a shunting locomotive has sent a wagon group to one handling station, the shunting locomotive does not need to go back to the marshalling yard before delivery of another wagon group. However, in a radial siding network, the shunting locomotive must return to the marshalling yard before running on to the next handling station. Furthermore, the hybrid siding network can be divided into some radial sub-branches, and each radial sub-branch is a branch-shaped siding network. In a hybrid siding network, after the
shunting locomotive has completed all the operations belonging to the same radial sub-branch, it must return to the marshalling yard before running onto another radial sub-branch. In a radial sub-branch, the working process of the shunting engine is the same as that of branch-shaped siding network. In this paper, we discuss optimization of simultaneous delivery and pickup wagon scheme on hybrid siding network and name this type of problem as SDPWS-HSN. Moreover, we develop a two-stage hybrid optimization procedure called as TSHOP.

The remainder of this paper is organized as follows. Section 2 of this paper discusses related earlier research efforts. Section 3 is devoted to the mathematical description of the SDPWS-HSN. In Section 4, we describe the mathematical formulation. Section 5 introduces a two-stage hybrid optimization procedure combining the three-phase approach and asynchronous iteration heuristic for solving SDPWS-HSN. The computational and compared experiments are described in Section 6, and the computational results show the effectiveness of the proposed method. The current work and extensions are summarized in the last section.

2. Literature Review

In this section, we review the related literature on simultaneous delivery and pickup wagons. Literature review indicates that relevant studies can be divided into three groups, that is, vehicle routing problem, vehicle assignment, and delivery and pickup problem. The focus of our literature review will primarily be on the model and approach since it is also what we have taken in this paper.


The dividing batch is the crucial point of the SDPWS-HSN and essentially belongs to the vehicle assignment. In recent years, many research studies have considered the vehicle assignment problem from different perspectives, using mathematical model method and heuristic algorithm. Choi et al. [13] considered the vehicle assignment problem with demand uncertainty in a hybrid hub-and-spoke network with a single hub. Spliet et al. [14] introduced the time window assignment vehicle routing problems with time-dependent travel times and developed an exact labeling algorithm and a tabu search heuristic. Györgyi et al. [15] proposed a probabilistic approach to solve the dynamic delivery and pickup problems with time window uncertainty. João et al. [16] presented an approach based on integrated flight scheduling and fleet assignment model to minimize generalized total transport network costs. Batsyn et al. [17] introduced the fleet site-dependent vehicle assignment problems with split deliveries and described two types of approaches. Wu et al. [18] presented a stated-choice survey to investigate the attractiveness to customers of two mechanisms for managing fleet volatility: virtual queuing and guaranteed advance reservation, and the results of the study are helpful for the manufacturer to classify the market. Li et al. [19] discussed a particular vehicle assignment problem arising when carrying out loading or unloading operations and proposed a bi-level programming model, and an alternating algorithm by updating predicted control parameters was given. Then, Li et al. [20] focused on the multistage heterogeneous fleet scheduling with fleet sizing decisions, established a mixed integer programming model, and developed a hybrid simulated annealing algorithm. Wang et al. [21] developed a collaborative neuro dynamic optimization approach to solve multivehicle task assignment problems. Qin et al. [22] proposed a dynamic task assignment method for vehicles based on the multiagent reinforcement learning. Zhilyakova et al. [23] presented graph methods for solving unconstrained and constrained locomotive assignment problem on a single-line railway section.

Many studies were addressed from the viewpoint of optimization models and solution methods for the delivery and pickup problem. The delivery and pickup problem has great reference significance for our research. Guo et al. [24] turned the delivery and pickup problem into a single machine scheduling problem and used Hungarian algorithm to calculate the optimal solution of the assignment problem. Adamo et al. [25] presented the problem of cargo pickup path and speed optimization with time window constraints and proposed a branch and bound algorithm. Wang et al. [30] introduced the problem of cooperative green delivery and pickup with the goal of minimizing carbon emissions. An exact solution procedure based on cooperative game
theory was provided. Ghilas et al. [27] proposed a branch and bound algorithm to solve the problem of delivery and pickup of cargo with time windows. Haddad et al. [31] discussed the multivehicle one-to-one delivery and pickup problem with split loads and proposed an iterated local search metaheuristic and a branch-and-price algorithm. Danloup et al. [32] presented the problem of delivery and pickup of cargo with transshipment and proposed a large-scale neighborhood search algorithm to solve the problem. Wang et al. [26] proposed a hybrid algorithm based on integrated improved particle swarm optimization algorithm and ant colony algorithm to solve the problem of delivery and pickup optimization based on shared fleet of collaboration enterprise. Capelle et al. [31] built an integer programming model considering the location-routing problem of cargo operations and provided a column-generation solving algorithm. Zhu et al. [32] proposed a synchronous delivery and pickup problem with stochastic demand and developed a new delivery strategy which allows vehicles to cooperate in the delivery of goods. Boysen et al. [33] described the operation process of railway container yard. Li et al. [34] proposed the optimization of cargo train inspection and disassembly, and assembly operations at the railway marshalling station and formulated a sorting model based on multimodulation and inspection groups. Shi et al. [35] built a multilayer network flow model to describe the train arrival, disintegration, assembly, grouping, and starting operations of railway marshalling stations and gave a heuristic solution. Jaehn et al. [36] presented the optimization of delivery and pickup wagons for railway sidings and introduced exact algorithms with polynomial runtimes. Gestrelius et al. [37] comprehensively analyzed the arrival, disintegration, assembly, marshalling, and departure process of freight trains in marshalling yard and gave an integer programming model of shunting operation in marshalling yard with the objective function of minimizing the number of wagons towed by the engine and the occupancy of shunting line.

The focus of this paper is the development of some models based on the SDPWS-HSN to aid in making decisions on the delivery and pickup scheme on hybrid siding network. Further, a two-stage hybrid optimization procedure (TSHOP) is proposed for solving the model. A three-phase approach as the first component of TSHOP is developed to generate initial solution. An asynchronous iteration heuristic as the second component of TSHOP is provided to update solution population. The conclusion of the experiment can intuitively provide help for decision makers of railway terminals in formulating development strategies.

3. Problem Formulation

The whole processes of delivery and pickup scheme of local wagons on hybrid siding network are described in this section. We explicate some foundational concepts and analyze the characteristics of the problem. Then, we define the relevant notation and formulate simultaneous delivery and pickup shunting scheme of wagons on hybrid siding network (SDPWS-HSN) as a programming model.

3.1. Understanding Delivery and Pickup Shunting of Wagons on Hybrid Siding Network. Here we discuss the layout of hybrid siding network which consists of marshalling yard and handling stations in railway terminal.

The marshalling yard consists of three kinds of working areas: the arrival yard, the departure yard, and the shunting yard which has some classification tracks with a sophisticated braking system. The arrival yard and shunting yard are often connected by a hump, and an automatic switching system sequentially directs the wagons to their assigned classification tracks of shunting yard. In handling stations, the local wagons can be loaded or unloaded.

Large quantities of inbound freight trains arriving to the marshalling yard in a nondirect manner are stopped on the arrival yard where local wagons are decoupled and inspected. Considering the relevant constraints, these local wagons need to be rolled into the shunting yard and sorted into their assigned classification tracks. When the delivery and pickup shunting of wagons is carried out, a shunting locomotive is sent from the engine depot and the local wagons on some given tracks are coupled. After all handling stations are visited, the taken-out wagons are pulled back to shunting yard and sorted into some new outbound trains. Finally, outbound trains must leave from the departure yard. The continuous delivery and pickup shunting begins. The process block diagram illustrating the delivery and pickup shunting of wagons on hybrid siding is shown in Figure 1.

3.2. Problem Definition and Notation. To formulate this problem, we present the complete notation for the problem here.

3.2.1. Sets. \( U \) is the set of marshalling yard and handling stations in hybrid siding network. Define \( U = \{ u | u = 0, 1, \ldots, n \} \), where \( n \) is the total number of handling stations. Especially, we use \( u = 0 \) to denote marshalling yard.

\( H \) is the set of radial branches in hybrid siding network. Define \( H = \{ h | h = 1, 2, \ldots, h \} \), where \( h \) is the total number of radial branches.

\( L \) is the set of inbound freight trains arriving at marshalling yard. Define \( L = \{ l | l = 1, \ldots, L \} \), where \( L \) is the total number of inbound freight trains throughout the shunting horizon. Let \( l \) denote the serial number of inbound freight train. Especially, we use \( l = 0 \) to denote wagon groups which are parked on handling station \( u \) at the beginning of shunting horizon.

\( S \) is the set of operation types. \( S \) is written as \( S = \{ s | s = 1, 2 \} \). We denote \( s = 1 \) as delivery wagons. Let \( s = 2 \) represent pickup wagons.

\( R \) is the set of local wagon groups. Define \( R = \{ r | r \in L, h \in H, u \in U \} \), which is decoupled from inbound freight train \( l \) to handling station \( u \) in the radial branch \( h \) for loading and unloading. Especially, we define \( r \in \Omega \) as the detention wagon groups which are parked on handling station \( u \) in the branch \( h \) at the beginning of shunting horizon.

\( \Omega \) is the set of local wagon groups. Define \( \Omega = \{ \Omega_s | s \in S \} \), where \( \Omega_s \) is the task of delivery wagon group \( \Omega_s \), \( \Omega_s (2) \) is the task of pickup wagon
group $l_{hu}$, and $0^h(2)$ is the task of pickup detention wagon group $l_{hu}$.

$W$ refers to the batch serial number sequences of delivery and pickup wagons. Define $W = \{w | w = 1, \ldots, \overline{w} \}$, where $\overline{w}$ is the total number of batches for the whole shunting horizon.

### 3.2.2. Parameters

$\pi_{u, u'}$ is the travel time from handling station $u$ to handling station $u'$.

$Z(l^h_u)$ is the number of local wagon group $l^h_u$. Especially, we use $Z(0^h_u)$ to denote the number of detention wagon groups at handling station $u$ in radial branch $h$.

$\pi(l^h_u)$ is the loading and unloading time of the wagon group $l^h_u$.

$\chi(l^h_u)$ is the completion time point of shunting for break-up of the inbound freight train $l$.

$[ET(l^h_u), LT(l^h_u)]$ is the departure time window of outbound train coupled by the local wagon group $l^h_u$. ET ($l^h_u$) is the earliest departure time point of outbound train coupled by the local wagon group $l^h_u$. LT ($l^h_u$) is the latest departure time point of outbound train coupled by the local wagon group $l^h_u$.

$d$ is the tonnage of shunting engine traction. Here it is expressed as the number of wagons pulled by the shunting engine.

$\eta_1$ is the unit-minute operating cost of the shunting engine.

$\eta_2$ is the unit-minute operating cost of the local wagon group.

$\eta_3$ is the unit-minute penalty cost caused by the failure to retrieve the local wagon group $l^h_u$ in time.

### 3.2.3. Activity Variables

$\pi_u$ is the time point of shunting engine’s departure from marshalling yard in the $u$th batch.

$\eta_u$ is the time point of shunting engine’s return to marshalling yard in the $u$th batch.

$\pi(l^h_u)$ is the time point of local wagon group $l^h_u$’s return to marshalling yard. $\eta(l^h_u)$ is the completion time point of handling operations of the local wagon group $l^h_u$.

$t(j)$ is the time when the local wagon group numbered $j$ is taking or sending.

$\chi_{l^h_u}$ is the number of wagon dispatched to handling station $u$ in the $u$th batch. $\chi_{l^h_u}$ is the number of wagon retrieved from handling station $u$ in the $u$th batch. $\chi_{l^h_u}$ is the total number of wagons dispatched to all handling stations in the $u$th batch. $\chi_{l^h_u}$ is the total number of wagons retrieved from all handling stations in the $u$th batch. $\chi_{l^h_u}$ is the total number of wagons pulled by the shunting engine between handling stations $u$ and $u'$ in the $u$th batch.

### 3.2.4. Decision Variables

$\lambda(l^h_u(s))$ is the batch serial number of task $l^h_u(s)$.

$\psi(l^h_u(s))$ is the access order sequence of task set $R_t$.

$\eta_{l^h_u}$ is the number of task $l^h_u$.

$\chi_{l^h_u}$ is the order number of task $l^h_u$.

$\nu(l^h_u(s))$ is the access order sequence including path $(u, u')$ in the $u$th batch, and 0 otherwise.

### 3.3. Objective Function

The objective function aims to minimize the total cost composed of wagons’ operating costs, shunting engines’ operating cost, and penalty cost. It is given by

$$
\text{Min} F = \sum_{w=1}^{\overline{w}} (\pi_w - \pi_{w'})(\eta_1 + \sum_{s=1}^{\overline{w}} \sum_{u=1}^{\overline{u}} \sum_{u'=1}^{\overline{u'}} \eta_{\lambda(s)} \psi_{\lambda(s)} \nu_{\lambda(s)} \nu_{\lambda(s)} \eta_2 + \sum_{l=0}^L \sum_{h=1}^{\overline{h}} \sum_{u=1}^{\overline{u}} \max (\pi(l^h_u) - \text{ET}(l^h_u), 0) \eta_3
$$

The first term in the objective function is the shunting engines’ operating cost. It can be obtained by the product of the time consumption of one batch operation and the unit-minute operating cost. The second term in objective function is the wagons’ operating cost. The value can be computed by the product of the wagons’ number of any path, travel time of any path, and the unit-minute operating cost. The third term in objective function is the penalty cost. It can be counted by the difference among the earliest departure time.
point of departure outbound train and retrieving time point of local wagon groups.

3.4. Constraints.

(1) Relationship between pickup wagon and delivery wagon: any local wagon group \( I^p_h \) includes delivery task \( I^d_h \) (1) and pickup task \( I^u_h \) (2). Obviously, the order number of delivery task must be less than that of pickup task. It can be written as

\[
\lambda(I^p_h(1)) < \lambda(I^u_h(2)), \forall I^u_h \in R. \tag{2}
\]

(2) Relationship between radial branch and operation batch: the local wagon groups going to different radial branch must not be divided into the same batch. It is given by

\[
w(I^p_h(s)) \neq w(I^u_h(s')), \forall I^p_h(s), I^u_h(s') \in R, h \neq h'. \tag{3}
\]

(3) Traction number of shunting engine: the total number of wagons pulled by the shunting engine in any path does not exceed the traction number of shunting engine. We have

\[
\chi^w_{u, u'} \leq d, \forall w \in W, u, u' \in U. \tag{4}
\]

(4) Relationship between task and operation batch: the task of delivery and pickup wagon can only be allocated to one batch number. It can be expressed as

\[
\sum_{w=1}^{\pi} w(I^p_h(s)) = 1, \forall I^p_h(s) \in R. \tag{5}
\]

(5) Equalization of wagon amount: the sum amount of delivery wagons and detention wagons is equal to the amount of pickup wagons over the entire planning horizon. It is shown as follows:

\[
\sum_{w=1}^{\pi} \chi^w_u + \sum_{h=1}^{\pi} \sum_{u=1}^{\pi} Z(O^u_h) = \sum_{w=1}^{\pi} \chi^w_u. \tag{6}
\]

(6) Marshalling yard as the start point and destination point of task: there is only one marshalling yard in the railway freight terminal studied in this paper. So, any task starts from the marshalling yard and eventually returns to the marshalling yard. We have

\[
y_{oww} = y_{oww} = 1, \forall u \in U, w \in W. \tag{7}
\]

(7) Relationship between local wagon groups’ retrieved time and outbound train departure time: the time point of local wagon group’s return to marshalling yard must be earlier than the latest departure time point of outbound train coupled by the local wagon group. It obviously can be expressed as

\[
\pi(I^p_h) \leq LT(I^d_h), \forall I^d_h \in R. \tag{8}
\]

(8) Handling station capacity: the wagons’ amount for loading and unloading in same station and batch must not exceed the capacity of handling station. We have

\[
Z(0^u_h) + \sum_{w=1}^{\pi} (\chi^w_u - \chi^w_u) \leq Z_u, \forall u \in U. \tag{9}
\]

3.5. Reconstruction of the Model. We simplify the mathematical model by reducing the constraint on the purpose of developing solving method. Constraints (8) and (9) are transformed into the penalty functions and then are put into the objective function as one component. So, the reformulated mathematical model is given by

\[
\min F_{\text{new}} = F + M_1 \sum_{h=1}^{\pi} \sum_{u=1}^{\pi} \max\{0, (\pi(I^p_h) - LT(I^d_h))\}
\]

\[
+ M_2 \sum_{w=1}^{\pi} \sum_{u=1}^{\pi} \max\left\{\left[ Z(0^u_h) + \sum_{w=1}^{\pi} (\chi^w_u - \chi^w_u) - Z_u \right], 0 \right\},
\]

s.t. constraints (2)–(7).

Here the parameters \( M_1 \) and \( M_2 \) are defined as penalty coefficients and are two large enough positive numbers. The reformulated objective function (10) is defined as measure function to evaluate the performance of the proposed approach.

4. Solution Algorithm

As an NP problem, it is difficult and inefficient to solve the SDPWS-HSN model by traditional methods. A novel method is proposed to solve this problem. This section proposes a two-stage hybrid optimization procedure, called as TSHOP, to solve the SDPWS-HSN. TSHOP includes two components: three-phase approach for initial solution and asynchronous iteration heuristic for updating solution population. Section 4.1 provides the three-phase approach used to code task sequence, divide task batch, and generate access order sequence population. Section 4.2 develops the asynchronous iteration heuristic used to make a decision on optimal routing of delivery and pickup wagons.

4.1. Initial Solution Generation with TPA. The three-phase approach composed of coding task sequence, dividing task batch, and generating access order sequence population is developed to generate initial solution. This approach is
abbreviated as TPA in order to facilitate the expression of the problem. An overview of the framework of TPA is shown in Figure 2.

Stage 1. Solution encoding.

Step 1.1. Natural number coding sequence.
According to the task set $\Omega$, natural number is used to mark all tasks $l_h^u(s)$ in ascending order. The delivery task $l_h^u(1)$ is coded as an odd number and the pickup task $l_h^u(2)$ is encoded as an even number. For the convenience of expression, let $\Omega$ be rewritten as $\Omega = \{l_h^u(s) + 1, \ldots, l_h^u(s) + Q | l \in L, h \in H, u \in U, s \in S\}$, and $Q$ is the total number of delivery and pickup local wagon groups. Define $\sigma(\Omega)$ as the natural number coding sequence of task set $\Omega$. We have $\sigma(\Omega) = \{\sigma(l_h^u(s) + 1), \ldots, \sigma(l_h^u(s) + Q)\}$.

Step 1.2. Rearranged coding sequence.
Generating random number is executed $Q$ times and $Q$ random numbers correspondingly replace $Q$ natural numbers in $\sigma(\Omega)$. Then, natural number coding sequence $\sigma(\Omega)$ is ranked in ascending order of random numbers. So, the rearranged coding sequence $\beta(\Omega)$ can be given by $\beta(\Omega) = \{\beta(l_h^u(s) + 1), \ldots, \beta(l_h^u(s) + Q)\}$.

Step 1.3. Updated coding sequence.
According to constraint (2) in the SDPWS-HSN model, all rearranged coding sequences are rearranged so that the delivery wagon task $l_h^u(1)$ is located in front of pickup wagon task $l_h^u(2)$ with the same inbound freight train $l$, handling station $u$, and radial branch $h$. Any pair of tasks’ failure to meet the above restriction in the rearranged coding sequence shall be interchanged. So, the updated coding sequence $\gamma(\Omega)$ can be obtained and is expressed by $\gamma(\Omega) = \{\gamma(l_h^u(s) + 1), \ldots, \gamma(l_h^u(s) + Q)\}$.

Stage 2. Batch division.

Step 2.1. Dividing batch with radiation branch.
According to constraint (3) in the SDPWS-HSN model, two adjacent tasks going to different radial branches cannot be allocated into the same batch. The batch breakpoint is located between two adjacent tasks going to different radiation branches. It means that these tasks before and after breakpoint are assigned to different batches.

Step 2.2. Dividing batch with traction number.
According to constraints (4) and (5) in the SDPWS-HSN model, the wagon amount allocated to shunting engine in a batch should be less than shunting engine traction $d$. The batch breakpoint needs to be inserted into the proper location of updated coding sequence. According to the order of updated coding sequence, the wagon amount of tasks is accumulated in turn until it reaches the shunting traction number. Accumulation process is stopped and batch breakpoint is put in the updated coding sequence.


Step 3.1. Access order coding sequence.
We, respectively, execute the procedure of dividing batch with radiation branch and traction number for updated coding sequence so that the access order coding sequence can be obtained.

Step 3.2. Access order coding sequence population.
Stage 1 program and stage 2 program are repeatedly executed $3$ times so that $3$ access order sequences are created. The $3$ access order coding sequences are collected to set up access order coding sequence population. The access order coding sequence population is shown in Figure 3.

4.2. Solution Population Updating with AIH. In this section, we propose an asynchronous iteration heuristic, embedding primary updating and senior updating for delivery and pickup routing, called as AIH-PU&SU. An overview of the framework of AIH is shown in Figure 4. The detailed steps are as follows.

Stage 1. Setting control parameters of iteration procedure.
Let the access order coding sequence population be initial iteration population. The access order coding sequence is made as the iteration object in the iteration procedure. The iteration object will be continually updated using the proposed approaches. The iteration population is composed of $3$ iteration objects. Let $R$ be the max number of iteration throughout iteration procedure. The $\xi$th iteration population is expressed as $\omega(\xi = 1, \ldots, R)$. The iteration object number in each iteration population is kept to $3$ for whole iteration horizon.

Stage 2. Primary updating iteration population with pending and remaining group.
Step 2.1. Splitting iteration object group in iteration population.

Any \( \gamma \)th iteration population throughout whole iteration horizon is divided into two groups, i.e., pending iteration object group \( \omega_{\text{remaining}} \) and remaining iteration object group \( \omega_{\text{remaining}} \). The number of iteration object in the two groups is the same. The pending iteration object group \( \omega_{\text{remaining}} \) is updated firstly with proposed approach. Then, the remaining iteration object group \( \omega_{\text{remaining}} \) is updated secondly with different proposed approach. Finally, two implement object groups are combined into a new iteration population.

Step 2.2. Updating pending implement object group with random vector group.

(1) Pending implement object group.
Half of the implement objects in \( \gamma \)th iteration population \( \omega_{\text{remaining}} \) are randomly chosen to set up pending implement object group. The pending implement object group is given by \( \omega_{\text{remaining}} = \{y_1(\Omega), \ldots, y_{\gamma/2}(\Omega)\} \).

(2) Random number vector group.
\( \Omega/2 \) random number vectors with the certain range from 0 to 1 are generated to mark all implement objects in the pending set. The random number vector group is denoted as \( \rho_{\text{pending}} = \{\rho(y_1), \ldots, \rho(y_{\gamma/2})\} \). The relationship between pending implement object group and random number vector group is shown in Figure 5.

(3) New random number vector group.
One vector \( \rho(y_\gamma) \) is randomly selected from random number vector group \( \rho_{\text{pending}} \). The \( \gamma \)th vector \( \rho(y_\gamma) \) in random number vector group and selected vector \( \rho(y_\gamma) \) are put into equation (11) to calculate \( \rho_{\text{new}} \). The new random number vector can be obtained by

\[
\rho_{\text{new}} = \rho_{i\beta} + \Phi(\rho_{i\beta} - \rho_{k\beta}). \tag{11}
\]

Here we put a disturbance parameter \( \Phi \) into equation (11). The disturbance parameter is a random number between 0 and 1. It intends to achieve a larger searching range.

When all vectors in random number vector group \( \rho_{\text{pending}} \) perform the above updating procedure, a new random number vector group \( \rho_{\text{new}} \) can be obtained. The updating procedure of new random number vector group is shown in Figure 6.

(4) Updating pending implement object group.
All random number vectors in new random number vector group \( \rho_{\text{new}} \) are ranked in ascending order of random number value. Each implement object matched with random number vector should be reordered accordingly in the pending implement object group. So, the new pending implement object group can be obtained. Then, each implement object is compared with that of...
pending implement object group. Then, the implement object with lower function value will be collected into the updating pending implement object group. The updating pending implement object group is written as \( \omega_{\text{pending}}^{t} = \{ y_1^{t, \text{updating}}(\Omega), \ldots, y_{\xi/2}^{t, \text{updating}}(\Omega) \} \). The hierarchical structure of this process is outlined in Figure 7.

**Step 2.3**. Updating remaining implement object group with accumulation probability and random vector group.

1. Remaining implement object group.
   - The remaining half of the implement objects in \( \xi \)th iteration population \( \omega_{\xi}^{t} \) are collected to set up remaining implement object group. The remaining implement object group is given by \( \omega_{\xi}^{t, \text{remaining}} = \{ y_{\xi/2+1}^{t}(\Omega), \ldots, y_{\xi}^{t}(\Omega) \} \).
2. Updating remaining implement object group with accumulation probability.
   - For a minimization problem, the solution with lower objective function value has better performance. The cumulative probability of updating pending implement object group is given by
     \[
     \rho(y^{t, \text{updating}}) = \frac{\sum_{i=1}^{j} (1/\gamma_{i}^{t, \text{updating}})}{\sum_{i=1}^{\xi/2} (1/\gamma_{i}^{t, \text{updating}})} \quad j = 1, \ldots, \frac{\xi}{2},
     \]
   - The remaining implement object \( y_{\xi/2+j}^{t}(\Omega) \) is taken out in turn from remaining group \( \omega_{\xi}^{t, \text{remaining}} \).
   - Meanwhile a random number \( \phi \) uniformly distributed between 0 and 1 is generated. If there is
     \[
     \phi \leq \rho(y_{\xi/2+j}^{t, \text{updating}}),
     \]
     let \( y_{\xi/2+j}^{t, \text{new}}(\Omega) = y_{\xi/2+j}^{t, \text{updating}}(\Omega) \).
   - Besides, if there is \( \rho(y_{\xi/2+j}^{t, \text{updating}}) < \phi \leq \rho(y_{\xi/2}^{t, \text{remaining}}) \), let \( y_{\xi/2+j}^{t, \text{new}}(\Omega) = y_{\xi/2}^{t, \text{remaining}}(\Omega) \).
   - The first updating remaining implement object group is written as \( \omega_{\xi}^{t, \text{remaining}} = \{ y_{\xi/2}^{t, \text{remaining}}(\Omega), \ldots, y_{\xi}^{t, \text{remaining}}(\Omega) \} \).

3. Updating remaining implement object group with random vector group.
   - The approach used in stage 2 is adopted to generate the updating remaining implement object group. The updating remaining implement object group is written as \( \omega_{\xi}^{t, \text{remaining}} = \{ y_{\xi/2}^{t, \text{remaining}}(\Omega), \ldots, y_{\xi}^{t, \text{remaining}}(\Omega) \} \).

**Stage 3.** Senior updating iteration population with Lévy flight searching.

**Step 3.1.** Generating random number vector group. The random number vectors with the certain range from 0 to 1 are generated to mark all implement objects in the primary updating iteration population \( \omega_{\xi}^{t, \text{primary}} \). The random number vector group is denoted as \( \rho(y_{\xi}^{t, \text{primary}}) = \{ \rho(y_{1}^{t, \text{primary}}), \ldots, \rho(y_{\xi}^{t, \text{primary}}) \} \). The implement object with the lowest function value is searched from the primary updating iteration population \( \omega_{\xi}^{t, \text{primary}} \). It is called as the local implement object and denoted as \( y_{\text{optimal}}^{t} \). The random number vector matched with the local optimal implement object can be called as local optimal random number vector and written as \( \rho(y_{\text{optimal}}^{t}) \).

**Step 3.2.** Updating random number vector group with Lévy flight searching. Lévy flight searching is used to update further the random vector group. The short searching step and long searching step are alternately adopted to avoid falling into the local optimal solution and explore the wider range of solution space in the approach. The \( i \)th
vector $\rho(\gamma_{\text{primary}}^i)$ in the random number vector group $\rho_{\text{primary}}^i$ and local optimal vector $\rho(\gamma_{\text{optimal}}^i)$ are put into Lévy flight searching equation (14) to calculate $i$th new random number vector. The new random number vector can be obtained by

$$
\rho_{ij}^\text{new} = \rho_{ij}^\text{primary} + \phi \cdot \text{Lévy} \cdot \left( \rho_{ij}^\text{primary} - \rho_{ij}^\text{optimal} \right) \quad (14)
$$

The parameter $\phi$ is called as searching step factor. It is a random number with the range from 0 to 1 and intends to achieve a larger searching range. The parameter Lévy is a random variable with Lévy distribution and can be obtained by

$$
\text{Lévy} = \frac{\mathcal{N}}{|\rho|^{1/2}} \quad (15)
$$

The random variable $\rho$ obeys Gaussian distribution with mean 0 and variance 1. The random variable $\mathcal{N}$ follows Gaussian distribution with mean 0. The variance of $\mathcal{N}$ can be written as $\text{Var}(\mathcal{N}) = \Gamma(1 + \varepsilon) \times \sin(\pi \varepsilon) / \Gamma(1 + \varepsilon) \times 2^{(\varepsilon + 1)/2}/\Gamma$, where $\Gamma$ is the gamma function.

When all vectors in random number vector group $\rho_{\text{primary}}^i$ perform the above updating procedure, a new random number vector group $\rho_{\text{new}}^i$ can be obtained. The updating procedure of new random number vector group is shown in Figure 8.

**Step 3.3. Achieving senior updating iteration population.**

All random number vectors in new random number vector group $\rho_{\text{new}}^i$ are ranked in ascending order of random number value. Each implement object matched with random number vector should be reordered accordingly in the primary updating iteration population $\omega_{\text{update}}^i$. So, the new iteration population can be obtained. Then, each implement object is compared with that of primary updating iteration population. Then, the implement object with lower function value will be collected into the senior updating iteration population. The senior updating iteration population is written as $\omega_{\text{senior}}^i = \{\gamma_{\text{primary}}^{i,1}, \ldots, \gamma_{\text{primary}}^{i,n}\}$. The hierarchical structure of this process is outlined in Figure 9.

**Stage 4. Finding global optimal implement object.**

**Step 4.1. Termination criterion.**

For the whole iteration horizon, repeat stage 2 to stage 3 until the iteration number arrives at $R$.

**5. Case Study**

In this section, we try to evaluate the performance of the TSHOP provided in previous section to solve the SDPWS-HSN by some traditional measures such as objective function and execution time. The scenarios for SDPWS-HSN are described in Section 5.1. The key control parameters of TSHOP are set in Section 5.2. In Section 5.3, many small-size and large-size contrast experiments are developed...
to test the performance of the TSHOP compared with CPLEX solver and some proposed swarm intelligence algorithms. Section 5.4 reports the relationship between the iteration number and the quality of the solution in three sets of experiments.

5.1. Scenario Settings of the Experiments. In order to design the experiments scheme, the scenarios settings must be firstly set up. We choose Zhengzhou railway terminal in China as the tested target object and generate the tested dataset. However, the collected data still have some inconsistencies caused by the total number of local wagons, the time interval of the inbound trains, wagon number of each wagon group being dispatched to various handling stations, loading or unloading time of wagon group being dispatched to various handling stations, and so on. Therefore, the data have to be cleaned.

A hybrid siding network consisting of 11 handling stations is shown in Figure 10. The travel time $f$ between a pair of handling stations is listed in Table 1 where 0 represents the marshalling station. The handling station capacity is affected by the effective length of the handling siding. The handling station capacity is listed in Table 2. During the whole shunting horizon, 4 freight trains arrive at the marshalling station one by one. Data of arrival train and wagon flow in railway terminal are displayed in Table 3.

5.2. Parameter Setting. The test work has been executed using a computer with Intel(R) Core(TM) i7-10510U CPU @ 1.80 GHz 2.30 GHz. The hybrid heuristic combining asynchronous iteration procedure TSHOP has been coded in Microsoft Visual C++.

Here we give unit-minute operating cost of the shunting engine $\eta_1 = 16$, unit-minute operating cost of the local wagon group $\eta_2 = 1.4$, unit-minute penalty cost caused by the failure to retrieve the local wagon group in time $\eta_3 = 9$, and the tonnage of shunting engine traction $d = 40$.

Three key parameters are closely related to the performance of TSHOP. The three parameters are, respectively, the iteration population size $\mathcal{S}$, the maximum number of iterations $\mathcal{R}$, and disturbance parameter $\varphi$. Here they are tested to achieve a better combination configuration. The testing scope of three parameters is listed in Table 4. We collected 27 sets of parameters from Table 4 and adopted the experiment scenario mentioned in Section 5.1 to test them. Each set is tested 10 times. Also, we use two measures of performance. The first one is the average value of the objective functions obtained by executing TSHOP approach 10 times. In order to facilitate representing the measure, it is abbreviated to OFV. The second measure of performance is the average CPU time to execute the TSHOP approach 10 times. The testing results of parameter combination are illustrated in Table 5.

In Table 5, all instances are executed with no CPU time limit imposed. The performance of TSHOP approach improves with $\mathcal{S}$ and $\mathcal{R}$ increasing. However, $\mathcal{R}$ has more influence on the average value of the objective functions and
I has an obvious impact on the CPU time. Moreover, the parameter $\phi$ needs to arrive at a balance value. SUKhe too high or too low value of $\phi$ affects the performance of solution. From the testing results, we find the best parameter set as $I = 50$, $R = 160$, and $\phi = 0.6$.

### 5.3. Performance Evaluation

#### 5.3.1. Performance for Small-Size Instance

To evaluate the performance of the proposed heuristic, we firstly give four sets of contrast experiments with 8 item tasks, 10 item tasks, 12 item tasks, and 14 item tasks using the same hybrid siding network composed of 11 handling stations shown in Figure 10. They all are small-size instances. Here the CPLEX solver can be used to solve them. The four sets of contrast experiments are oriented to compare the performances of the TSHOP approach with that of CPLEX solver. The computer execution time of TSHOP approach and CPLEX solver is strictly limited within 3000 seconds. Each contrast experiment is executed 10 times. We use two measures to evaluate the performance. SUKhey are, respectively, the average objective function value and average CPU time. SUKhen, in order to facilitate the analysis of the problem, we provide the deviation rate as one analytical indicator. The deviation rate is abbreviated as RAT. It represents the deviation rate from TSHOP to CPLEX on optimal solution and can be calculated by

$$RAT = \frac{[OFV_{(TSHOP)} - OFV_{(CPLEX)}]}{OFV_{(CPLEX)}} \times 100\%.$$  

The results of performance comparison are displayed in Table 6.

### Table 3: Data of arrival train and wagon flow in railway terminal.

<table>
<thead>
<tr>
<th>Natural number coding of task</th>
<th>Serial number of train</th>
<th>Completion time point of train break-up</th>
<th>Handling stations</th>
<th>Number of local wagon group</th>
<th>Operation types</th>
<th>Handling time</th>
<th>Completion time point of handling operations</th>
<th>Departure time window of outbound train coupled by the local wagon group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>06:00</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>55</td>
<td>—</td>
<td>[9:20, 10:00]</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>06:00</td>
<td>1</td>
<td>11</td>
<td>2</td>
<td>55</td>
<td>—</td>
<td>[9:20, 10:00]</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>06:00</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>06:00</td>
<td>3</td>
<td>12</td>
<td>2</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>06:00</td>
<td>6</td>
<td>15</td>
<td>1</td>
<td>75</td>
<td>—</td>
<td>[12:00, 13:00]</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>06:00</td>
<td>6</td>
<td>15</td>
<td>2</td>
<td>75</td>
<td>—</td>
<td>[12:00, 13:00]</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>06:00</td>
<td>8</td>
<td>11</td>
<td>1</td>
<td>55</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>06:00</td>
<td>8</td>
<td>11</td>
<td>2</td>
<td>55</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>06:26</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>40</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>06:26</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>40</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>06:26</td>
<td>4</td>
<td>10</td>
<td>1</td>
<td>50</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>06:26</td>
<td>4</td>
<td>10</td>
<td>2</td>
<td>50</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>06:26</td>
<td>7</td>
<td>12</td>
<td>1</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>06:26</td>
<td>7</td>
<td>12</td>
<td>2</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>06:26</td>
<td>9</td>
<td>15</td>
<td>1</td>
<td>75</td>
<td>—</td>
<td>[11:40, 12:40]</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>06:26</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td>75</td>
<td>—</td>
<td>[11:40, 12:40]</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>07:30</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>45</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>07:30</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>45</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>07:30</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>35</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>07:30</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>35</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>07:30</td>
<td>10</td>
<td>13</td>
<td>1</td>
<td>65</td>
<td>—</td>
<td>[11:00, 12:10]</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>07:30</td>
<td>10</td>
<td>13</td>
<td>2</td>
<td>65</td>
<td>—</td>
<td>[11:00, 12:10]</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>08:10</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>30</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>08:10</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>30</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>25</td>
<td>4</td>
<td>08:10</td>
<td>6</td>
<td>12</td>
<td>1</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>26</td>
<td>4</td>
<td>08:10</td>
<td>6</td>
<td>12</td>
<td>2</td>
<td>60</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>27</td>
<td>4</td>
<td>08:10</td>
<td>10</td>
<td>15</td>
<td>1</td>
<td>75</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>08:10</td>
<td>10</td>
<td>15</td>
<td>2</td>
<td>75</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>08:10</td>
<td>11</td>
<td>9</td>
<td>1</td>
<td>45</td>
<td>—</td>
<td>[11:30, 12:00]</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>08:10</td>
<td>11</td>
<td>9</td>
<td>2</td>
<td>45</td>
<td>—</td>
<td>[11:30, 12:00]</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>—</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>35</td>
<td>06:30</td>
<td>—</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>—</td>
<td>6</td>
<td>14</td>
<td>2</td>
<td>70</td>
<td>07:00</td>
<td>—</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>—</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td>75</td>
<td>07:00</td>
<td>[10:20, 11:00]</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>—</td>
<td>11</td>
<td>8</td>
<td>2</td>
<td>40</td>
<td>06:20</td>
<td>—</td>
</tr>
</tbody>
</table>

### Table 4: Testing scope of three parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration population size $\mathcal{S}$</td>
<td>${50, 80, 100}$</td>
</tr>
<tr>
<td>Iteration maximum number $\mathcal{R}$</td>
<td>${100, 120, 160}$</td>
</tr>
<tr>
<td>Disturbance parameter $\phi$</td>
<td>${0.3, 0.6, 0.9}$</td>
</tr>
</tbody>
</table>

$\mathcal{S}$ has an obvious impact on the CPU time. Moreover, the parameter $\phi$ needs to arrive at a balance value. The too high or too low value of $\phi$ affects the performance of solution. From the testing results, we find the best parameter set as $\mathcal{S} = 50$, $\mathcal{R} = 160$, and $\phi = 0.6$.  

---

Table 3: Data of arrival train and wagon flow in railway terminal.

Table 4: Testing scope of three parameters.
For the first three sets of contrast experiments with 8 item tasks, 10 item tasks, and 12 item tasks, the objective function value of CPLEX solver is slightly better than that of TSHOP. But the CPU time of CPLEX solver is obviously higher than that of TSHOP. When the contrast experiments arrive at 14 item tasks, the CPU time of CPLEX solver goes further away from the reasonable interval. SUKhe feasible solution cannot be found by CPLEX solver within 3000 seconds and is denoted as the symbol "." SUKhus, one obvious conclusion is that the performance of TSHOP is significantly prominent when compared to CPLEX solver.

5.3.2. Performance for Large-Size Instance. When the increase of task number goes a step further, the CPU time of CPLEX solver will go further away from the reasonable interval. The swarm intelligence algorithm has outstanding performance advantages in solving large-size instance. In many swarm intelligence algorithms, the genetic algorithm (GA) and artificial bee colony algorithm (ABCA) have the characteristics of few parameters, strong search ability, and easy implementation. GA and ABCA can only be adopted to implement the iterative procedure in order to find optimal solution. Also, TPA procedure in TSHOP needs to be introduced to generate initial solution. The TPA procedure is drawn into GA and ABCA to generate two new approaches written as TPA-GA and TPA-ABCA. The two approaches will be compared with TSHOP approach proposed in this paper.

In this section, the three sets of large-size contrast experiments are oriented to compare the performances of the TSHOP approach with the TPA-GA approach and TPA-ABCA approach. In the first set of contrast experiment, there are 16 item tasks for delivery and pickup local wagon group. In the second set of contrast experiment, there are 24 tasks. In the third set of contrast experiment, the task amount increases to 34.

We use similarly two measures to evaluate the performance. The first measure of performance is the average objective function value obtained by the TSHOP approach, the TPA-GA approach, and TPA-ABCA approach. The second is the CPU time to run TSHOP program, TPA-GA program, and TPA-ABC program.

In order to facilitate the analysis of the problem, we present two analytical indicators. The first indicator denoted as RAT1 represents the deviation rate from the TSHOP to TPA-GA on the optimal solution. Let OFV (TPA-GA) be the average value of objective functions obtained by running TPA-GA approach 10 times. Let OFV (TSHOP) be the average value of objective functions obtained by running TSHOP approach 10 times. RAT1 is calculated by

\[ \text{RAT1} = \frac{\text{OFV(TPA-GA)} - \text{OFV(TSHOP)}}{\text{OFV(TSHOP)}} \times 100\% \]

The second indicator denoted as RAT2 represents the deviation rate from the TSHOP to TPA-ABC on the optimal solution. Let OFV (TPA-ABC) be the average value of objective functions obtained by running TPA-ABC approach 10 times. RAT2 is calculated by

\[ \text{RAT2} = \frac{\text{OFV(TPA-ABC)} - \text{OFV(TSHOP)}}{\text{OFV(TSHOP)}} \times 100\% \]

Here we chose control parameter combination from Table 4 for testing three compared algorithms. Each instance is solved 10 times using the parameter listed in Table 4, and the average results are recorded in Tables 7–9.

From above three sets of testing instances, we can conclude as follows:

1. For the three sets of different size instance, the performance of TSHOP is significantly prominent when compared to TPA-GA and TPA-ABC. Although the average CPU time of TSHOP is more than that of TPA-ABC and TPA-GA, the performance of TSHOP is better than others when the major criterion OFV is adopted. The average deviation rate from the TSHOP to TPA-ABC and TPA-GA on the optimal solution, respectively,
reaches 16% and 22%. The performance comparisons are shown in Table 10. In sum, we can conclude that the TSHOP is effective to obtain the good delivery and pickup wagon scheme for SDPWS-HSN problem.

(2) At the instance with 16 item tasks stage, it can be observed that the TSHOP approach exhibits slight superiority on the major criterion OFV and analytical indicators RAT. When the increase of task amount goes a step further into 24 item tasks stage, it is an interesting discovery that the superiority shows a steady upward trend. After entering the stage of 34 item tasks, an obvious observation is that the superiority begins to appear a accelerated upward trend. Meanwhile, the CPU time for obtaining solutions by TSHOP approach is still maintained in a reasonable range.

Now for the largest-size instance with 34 item tasks, we find the optimal implement objects, i.e., the optimal handling station accessing sequences. The TSHOP, TPA-ABCA, and TPA-GA are all executed by the best parameter set as $\Xi = 50$, $R = 160$, and $\phi = 0.6$. The result is listed in Table 11.

We focus on the performance of TPA-GA, TPA-ABCA, and TSHOP program under 160 iterations. Here the objective function value of global optimal implement objects obtained by the TPA-GA program executed with 160 iterations is 64074.6. The objective function value of global optimal implement objects obtained by the TPA-ABCA program executed with 160 iterations is 58081.8. The objective function value of global optimal implement objects obtained by the TSHOP program executed with 160 iterations is 46528.6. Then, the objective function values obtained by three approaches for solving the SDPWS-HSN problem are compared. The objective function values obtained by TSHOP program are obviously better than those solved by TPA-GA program and TPA-ABCA program.

5.4. Correlation between Solution Performance and Iteration Number. Compared with iteration population size $\Xi$ and disturbance parameter $\phi$, the iteration number $R$ has more influence on the solution performance. To further study the relationship between solution performance and iteration

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>OFV</th>
<th>CPU time (s)</th>
<th>RAT1 (%)</th>
<th>RAT2 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Xi$</td>
<td>$R$</td>
<td>$\phi$</td>
<td>TSHOP</td>
<td>TPA-GA</td>
</tr>
<tr>
<td>50</td>
<td>120</td>
<td>0.6</td>
<td>14246.94</td>
<td>16077.18</td>
</tr>
<tr>
<td>50</td>
<td>160</td>
<td>0.6</td>
<td>14047.16</td>
<td>16380.32</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>0.6</td>
<td>16978.24</td>
<td>19411.92</td>
</tr>
<tr>
<td>100</td>
<td>160</td>
<td>0.6</td>
<td>16601.68</td>
<td>19522.2</td>
</tr>
<tr>
<td>Average</td>
<td>15468.50</td>
<td>17847.9</td>
<td>16857.33</td>
<td>283.47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>OFV</th>
<th>CPU time (s)</th>
<th>RAT1 (%)</th>
<th>RAT2 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Xi$</td>
<td>$R$</td>
<td>$\phi$</td>
<td>TSHOP</td>
<td>TPA-GA</td>
</tr>
<tr>
<td>50</td>
<td>120</td>
<td>0.6</td>
<td>23829.40</td>
<td>27264.20</td>
</tr>
<tr>
<td>50</td>
<td>160</td>
<td>0.6</td>
<td>22288.18</td>
<td>27085.66</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>0.6</td>
<td>26591.74</td>
<td>31805.48</td>
</tr>
<tr>
<td>100</td>
<td>160</td>
<td>0.6</td>
<td>25677.24</td>
<td>32238.22</td>
</tr>
<tr>
<td>Average</td>
<td>24596.64</td>
<td>29598.39</td>
<td>29130.39</td>
<td>410.54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>OFV</th>
<th>CPU time (s)</th>
<th>RAT1 (%)</th>
<th>RAT2 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Xi$</td>
<td>$R$</td>
<td>$\phi$</td>
<td>TSHOP</td>
<td>TPA-GA</td>
</tr>
<tr>
<td>50</td>
<td>120</td>
<td>0.6</td>
<td>39470.56</td>
<td>56531.44</td>
</tr>
<tr>
<td>50</td>
<td>160</td>
<td>0.6</td>
<td>39011.44</td>
<td>55818.16</td>
</tr>
<tr>
<td>100</td>
<td>120</td>
<td>0.6</td>
<td>45658.96</td>
<td>64629.40</td>
</tr>
<tr>
<td>100</td>
<td>160</td>
<td>0.6</td>
<td>43847.14</td>
<td>64231.22</td>
</tr>
<tr>
<td>Average</td>
<td>41997.03</td>
<td>60302.56</td>
<td>53915.39</td>
<td>682.63</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance with 16 item tasks</th>
<th>Instance with 24 item tasks</th>
<th>Instance with 34 item tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAT1 (%)</td>
<td>0.15</td>
<td>0.20</td>
</tr>
<tr>
<td>RAT2 (%)</td>
<td>0.09</td>
<td>0.18</td>
</tr>
</tbody>
</table>
Table 11: The optimal solution for TPA-GA, TPA-ABC, and TSHOP applied to the instance with 34 item tasks.

<table>
<thead>
<tr>
<th>Natural number coding of task</th>
<th>Access order coding sequences</th>
<th>Handling station accessing sequences</th>
<th>Completion time of task</th>
<th>Batch serial number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TPA-GA</td>
<td>TPA-ABCA</td>
<td>TSHOP</td>
<td>TSHOP</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>29</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>31</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>25</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>30</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>22</td>
<td>20</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>1</td>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>26</td>
<td>32</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>15</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>16</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>22</td>
<td>34</td>
<td>29</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>19</td>
<td>13</td>
<td>30</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>19</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>30</td>
<td>20</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>17</td>
<td>33</td>
<td>5</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>18</td>
<td>16</td>
<td>6</td>
<td>23</td>
<td>9</td>
</tr>
<tr>
<td>19</td>
<td>27</td>
<td>7</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>28</td>
<td>8</td>
<td>31</td>
<td>10</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>9</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>13</td>
<td>4</td>
<td>31</td>
<td>7</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>23</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>12</td>
<td>24</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>17</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>14</td>
<td>33</td>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>28</td>
<td>17</td>
<td>26</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>29</td>
<td>18</td>
<td>27</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>28</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>10</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>33</td>
<td>7</td>
<td>18</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>34</td>
<td>8</td>
<td>14</td>
<td>26</td>
<td>8</td>
</tr>
</tbody>
</table>

![Figure 11: Continued.](a)
number, TSHOP program, TPA-GA program, and TPA-ABCA program are all set up with the same default parameter setting ($T = 50, \phi = 0.6$) and tested by the three sets of different size instances. The performance is shown in Figure 11. It can be seen that the superiority of TSHOP appears an accelerated upward trend along with the increasing of iteration times.

6. Conclusions

This study aims to develop a model and approach to aid in making decisions on delivery and pickup scheme of wagons on the hybrid siding network of the railway terminal. We name the problem as simultaneous delivery and pickup wagon scheme on hybrid siding network in railway terminal (SDPWS-HSN). In this paper, a mathematical programming model has been discussed for SDPWS-HSN. As an NP problem, it is difficult and inefficient to solve SDPWS-HSN model by using traditional methods. A novel heuristic algorithm is proposed to address this problem.

In this paper, we propose a two-stage hybrid optimization procedure, abbreviated as TSHOP, to solve the SDPWS-HSN. Firstly, a three-phase approach composed of coding task sequence, dividing task batch, and generating access order sequence population is developed to generate initial solution. Secondly, the asynchronous iteration heuristic is provided to update the solution population. Finally, we evaluate the performance of the TSHOP in terms of two traditional measures: objective function and execution time. The results show that the proposed heuristic algorithm is able to obtain better solutions with reasonable CPU time.

Future research will focus on some new heuristics to solve the SDPWS-HSN. Another prospective research will be the extension of the SDPWS-HSN model by considering some further realistic aspects and relaxing some hard constraints.

Data Availability

All data, models, and codes generated or used during the study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the Philosophy and Social Science Applied Research Major Project of Henan Province Education Department of China (grant no. 2022-YYZD-24), Henan Province Philosophy and Social Science Planning Project (grant no. 2021BJJ087), Henan Province Science and Technology Research Plan Project of China (grant no. 202102310310), and National Natural Science Foundation of China (grant nos. U1604150 and U1804151).

References


